Report on MovieLens and Recommendation System

HarvardX Data Science Capstone Project

Raphael Kummer

2023-03-08

Contents

MovieLens

Introduction

This is a report on the MovieLens data analysis and a recommendation model training and the models achieved performance. First the dataset is to be explored, possibly cleaned and inspected to evaluate possible training approaches. Next part is building a ML model to recommend movies to users.

Dataset

Grouplens created a movie rating dataset. The 10M dataset (Harper and Konstan 2015) used in this project is a subset of 10 million ratings of 10'000 movies by 72'000 random selected users.

Initial setup

Given is the loading of the MovieLens 10M dataset, split into an *edx* and a *final_holdout_test* set containing 10% of the MovieLens data only used for validating at the end. The dataset contains userId, movieId, rating, timestamp, title, and genre.

Goal

This dataset is used to explore and gain insight on how an effective recommendation algorithm could be developed. Such a machine learning algorithm is then developed and tested against the *final_holdout_test* set.

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## Loading required package: caret
##
## Loading required package: lattice
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

Analysis

Data Inspection and preprocessing

```
## Loading required package: devtools
```

```
## Loading required package: usethis
```

```
## Loading required package: benchmarkme
```

```
## Loading required package: rmarkdown
```

```
## Loading required package: lubridate
```

```
## Loading required package: timechange
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
## Loading required package: scales
```

```
##
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## Loading required package: parallel
```

First lets take a closer look at the *edx* dataset structure and some of its content.

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

There are several columns in this dataset: - userId: an identifier for the individual user who rated the movie. - movieId: an identifier for movie that was rated. - rating: a rating that was given for this movie (from that user). The scale of the ratings is yet to be identified. - timestamp: a timestamp in UNIX format - title: the title of the movie including the year of release - genres: a list of genres associated with this movie. Multiple genres are separated by '|'.

There are several aspects of the edx dataset to consider exploring: - Several movies may be rated way above average because of a very good story or production. - Some genres (or genre combinations) may be rated higher than others - User ratings are possibly biased or generally rated higher or lower. - User ratings in relation to a genre. User A maybe likes Horror and Action but rates movies of other genres typically lower - User ratings in relation to movie release year - User ratings in relation to popularity of movies (indie vs blockbuster). And probably lots more.

Check if there are any NA in the dataset

```
anyNA(edx)
```

```
## [1] FALSE
```

Lets list all genres. We notice a unusual genre named (no genres listed). On further investigation, only one movie (PUll My Daisy) has no genre listed. According to IMDB the genre of this movie from 1958 is "Short". Let's separate the genres listed and add the first three of each movie into separate columns.

```
# List genres
unique(unlist(strsplit(edx$genres, "\\|")))
```

```
##  [1] "Comedy"             "Romance"            "Action"
##  [4] "Crime"              "Thriller"           "Drama"
##  [7] "Sci-Fi"             "Adventure"          "Children"
## [10] "Fantasy"            "War"                "Animation"
## [13] "Musical"            "Western"            "Mystery"
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

```
# inspect "(no genres listed)" movies
subset(edx, genres=="(no genres listed)")
```

	userId	movieId	rating	timestamp	title	genres
1025055	7701	8606	5.0	1190806786	Pull My Daisy (1958)	(no genres listed)
1453345	10680	8606	4.5	1171170472	Pull My Daisy (1958)	(no genres listed)
4066835	29097	8606	2.0	1089648625	Pull My Daisy (1958)	(no genres listed)
6456906	46142	8606	3.5	1226518191	Pull My Daisy (1958)	(no genres listed)
8046611	57696	8606	4.5	1230588636	Pull My Daisy (1958)	(no genres listed)
8988750	64411	8606	3.5	1096732843	Pull My Daisy (1958)	(no genres listed)
9404670	67385	8606	2.5	1188277325	Pull My Daisy (1958)	(no genres listed)

```r
# "Pull My Daisy" from 1958 is the only movie without a genre. According to IMDB its genre is "Short".
edx$genres <- ifelse(edx$genres == "(no genres listed)", "Short", edx$genres)

# Extract first few listed genres per movie in separate columns
edx$main_genre  <- sapply(strsplit(edx$genres, "\\|"), function(x) x[1])
edx$side1_genre <- sapply(strsplit(edx$genres, "\\|"), function(x) x[2])
edx$side2_genre <- sapply(strsplit(edx$genres, "\\|"), function(x) x[3])
```

Then we transform the UNIX timestamps to date/time and for ease of use also extract the year the movie was rated. And separate the release year, embedded in the title, for possible further investigation.

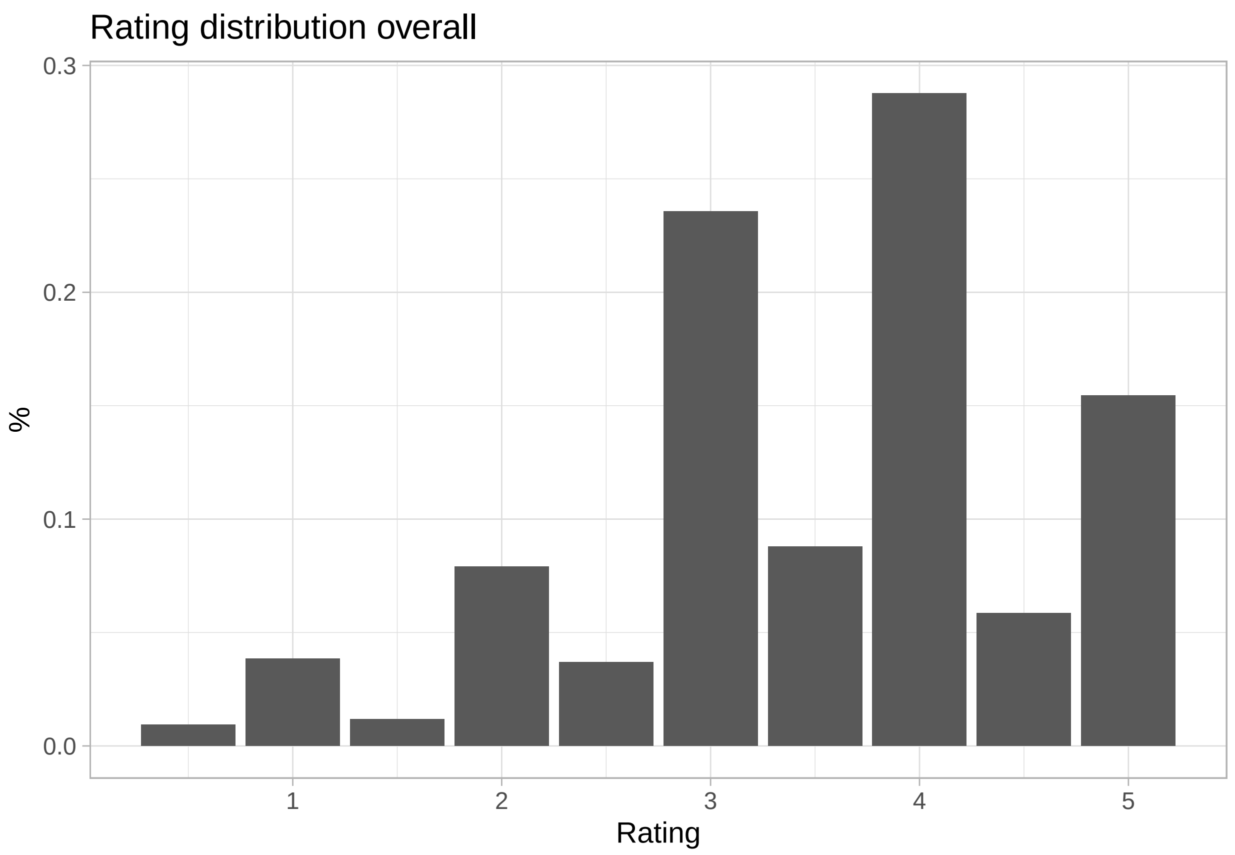
```r
# Extract Rating Year
edx <- edx %>%
  mutate(date=as_datetime(timestamp), yearrated=year(date))

# Extract Release Year
edx <- edx %>%
  mutate(releaseyear=as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"), regex("\\d{4}"))))
```

There are ….. unique genres in the dataset.

Lets plot some distributions, starting the distribution of ratings.

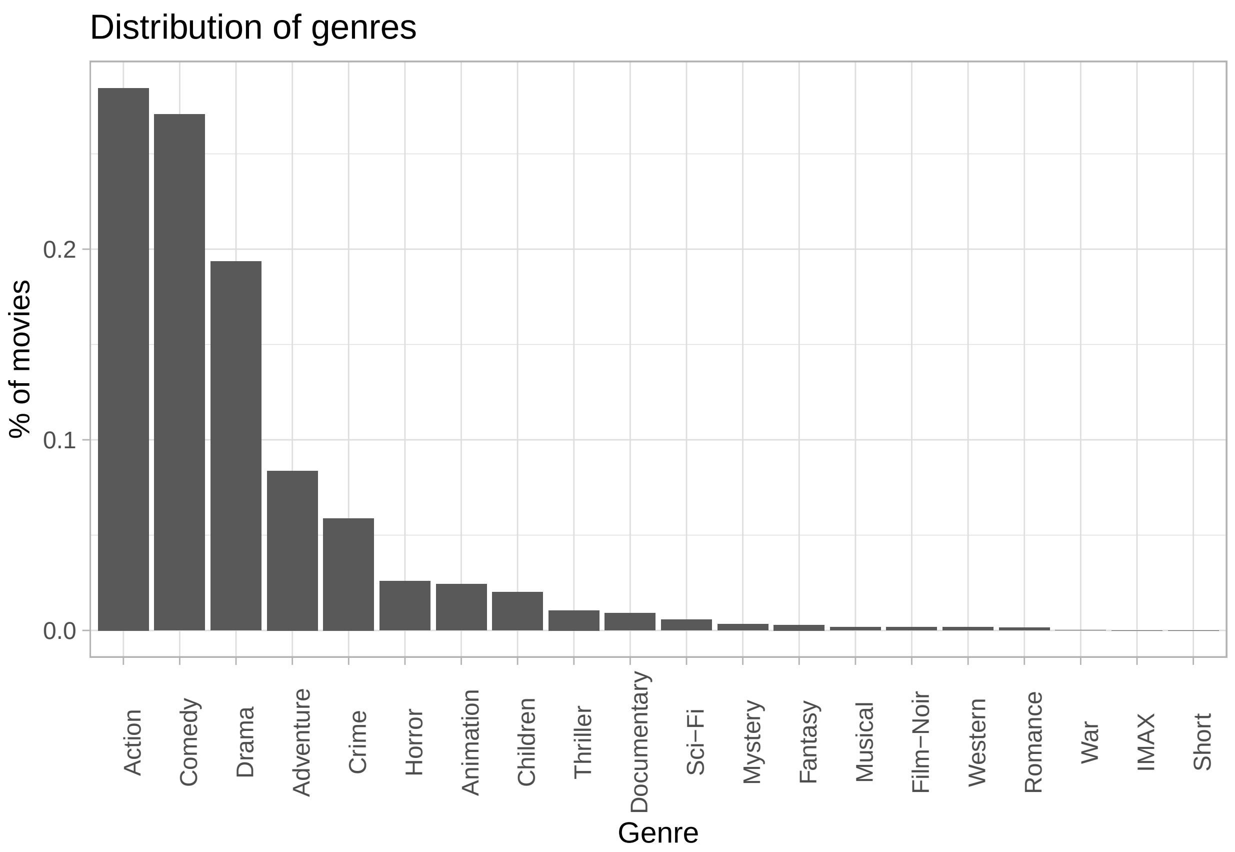
```r
# Distribution of ratings
edx %>%
  group_by(rating) %>%
  summarise(perc = n()/nrow(edx)) %>%
  ggplot(aes(rating, perc)) +
    geom_col() +
    theme_light() +
    labs(x="Rating", y="%", title = "Rating distribution overall")
```



Most ratings are given as full number ratings (4, 3 or 5). Also the half-point ratings distribution follows a similar distribution as the full number ratings.

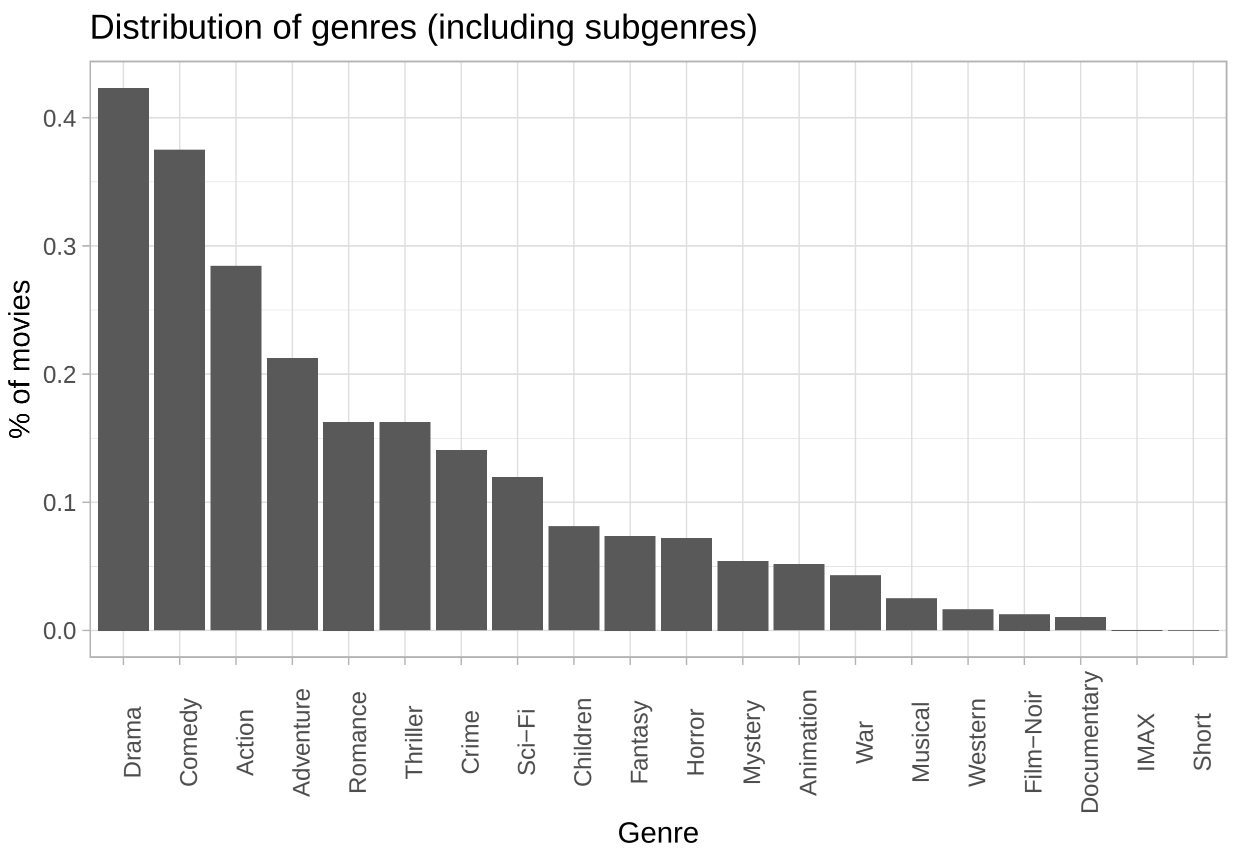
Now the distribution of the genres.

```
# Distribution of genres
edx %>%
  group_by(main_genre) %>%
  summarise(perc = n()/nrow(edx)) %>%
  ggplot(aes(reorder(main_genre, -perc), perc)) +
    geom_col() +
    theme_light() +
    theme(axis.text.x = element_text(angle=90)) +
    labs(x="Genre", y="% of movies", title = "Distribution of genres")
```



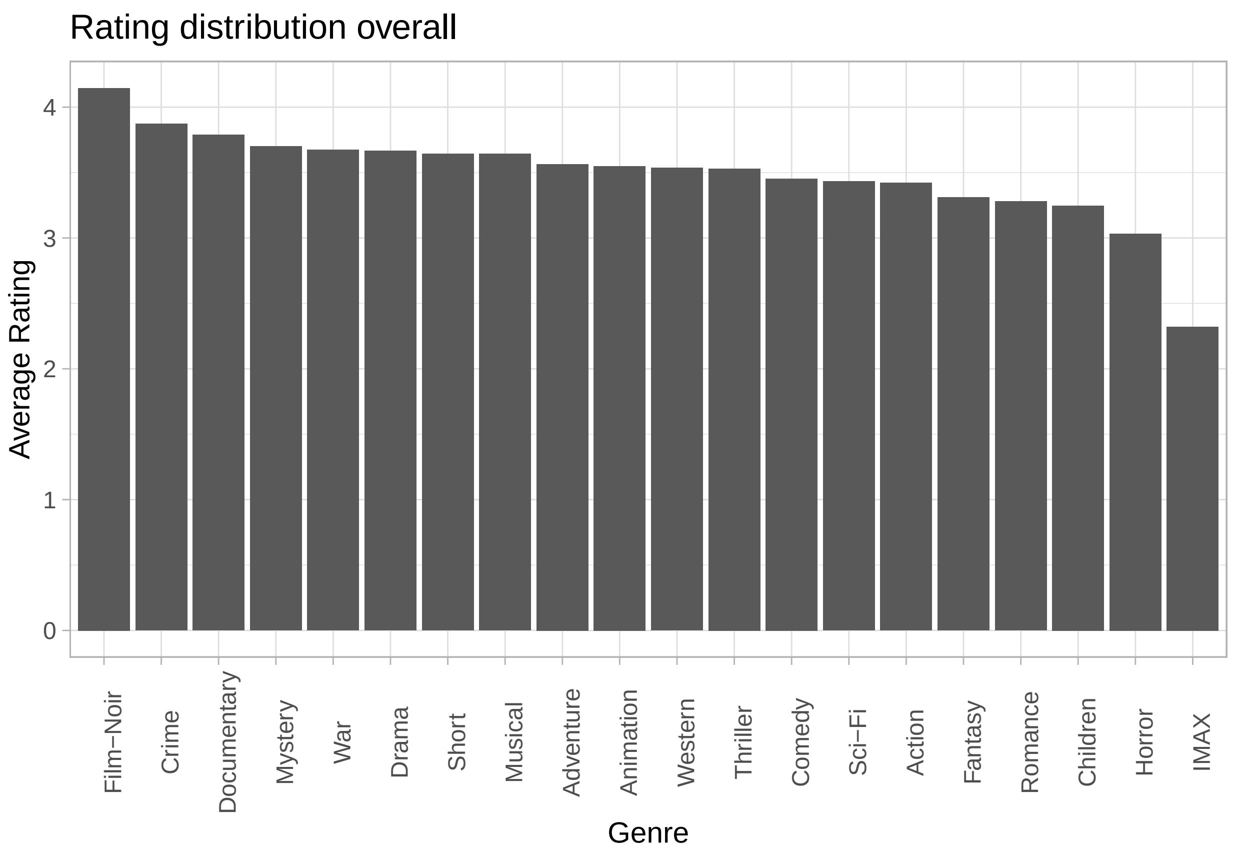
Most genres listed are Action, Comedy and Drama. What about when taking the sub genres into account.

```
edx %>%
  select(main_genre, side1_genre, side2_genre) %>%
  pivot_longer(cols=c(main_genre, side1_genre, side2_genre), values_to = "genre", values_drop_na = TRUE
  group_by(genre) %>%
  summarise(perc = n()/nrow(edx)) %>%
  ggplot(aes(reorder(genre, -perc), perc)) +
    geom_col() +
    theme_light() +
    theme(axis.text.x = element_text(angle=90)) +
    labs(x="Genre", y="% of movies", title = "Distribution of genres (including subgenres)")
```



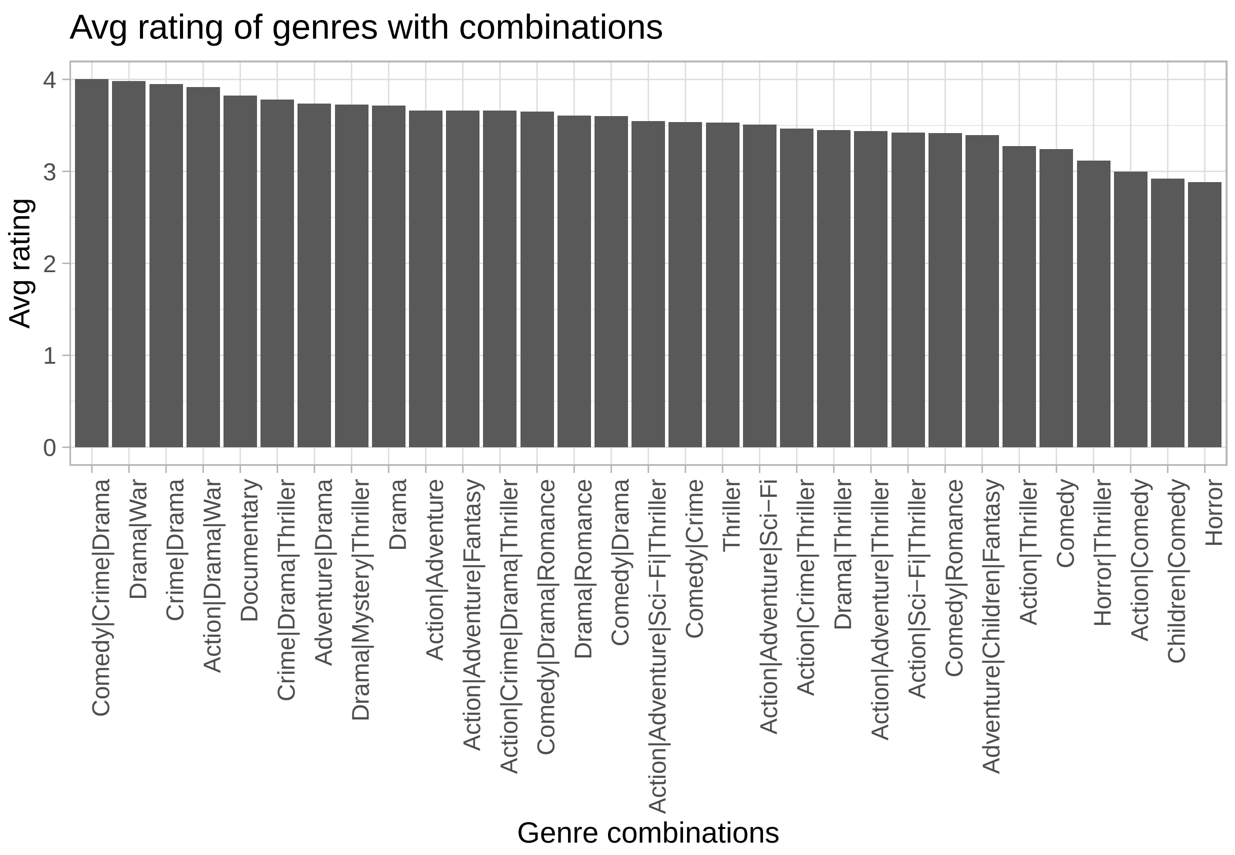
When second and third listed genres are taken into account, the distribution is much finer. Top genres are still Drama, Comedy and Action but positions changed slightly. Drama is therefore an often used side genre, e.g. in combination with Romance, Thriller or others.

```r
# Ratings per genre
edx %>%
  group_by(main_genre) %>%
  summarise(avg_genre_rating = mean(rating)) %>%
  arrange(desc(avg_genre_rating)) %>%
  ggplot(aes(reorder(main_genre, -avg_genre_rating), avg_genre_rating)) +
    geom_col() +
    theme_light() +
    theme(axis.text.x = element_text(angle=90)) +
    labs(x="Genre", y="Average Rating", title = "Rating distribution overall")
```



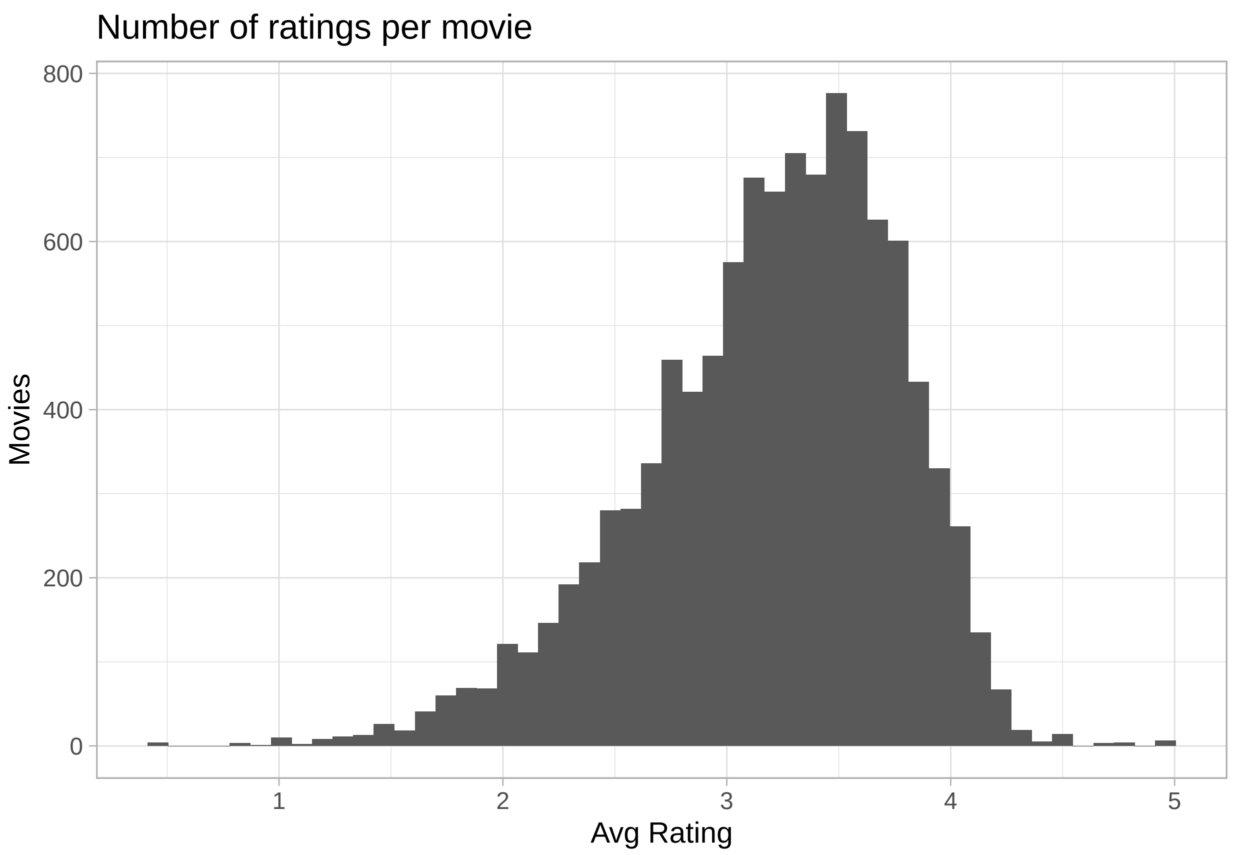
```
# Ratings of genres show "intellectual" movie genres (e.g. Film-Noir, Crime, Drama..) are rated higher
# associated with entertainment (e.g. Action, Fantasy, Horror)
```

```
# Average ratings of genre combinations with more than 50000 ratings
edx %>%
  group_by(genres) %>%
  summarize(n=n(), avg=mean(rating)) %>%
  filter(n >= 50000) %>%
  mutate(genres = reorder(genres, -avg)) %>%
  ggplot(aes(x=genres, y=avg)) +
    geom_col() +
    theme_light() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    labs(x="Genre combinations", y="Avg rating", title="Avg rating of genres with combinations")
```



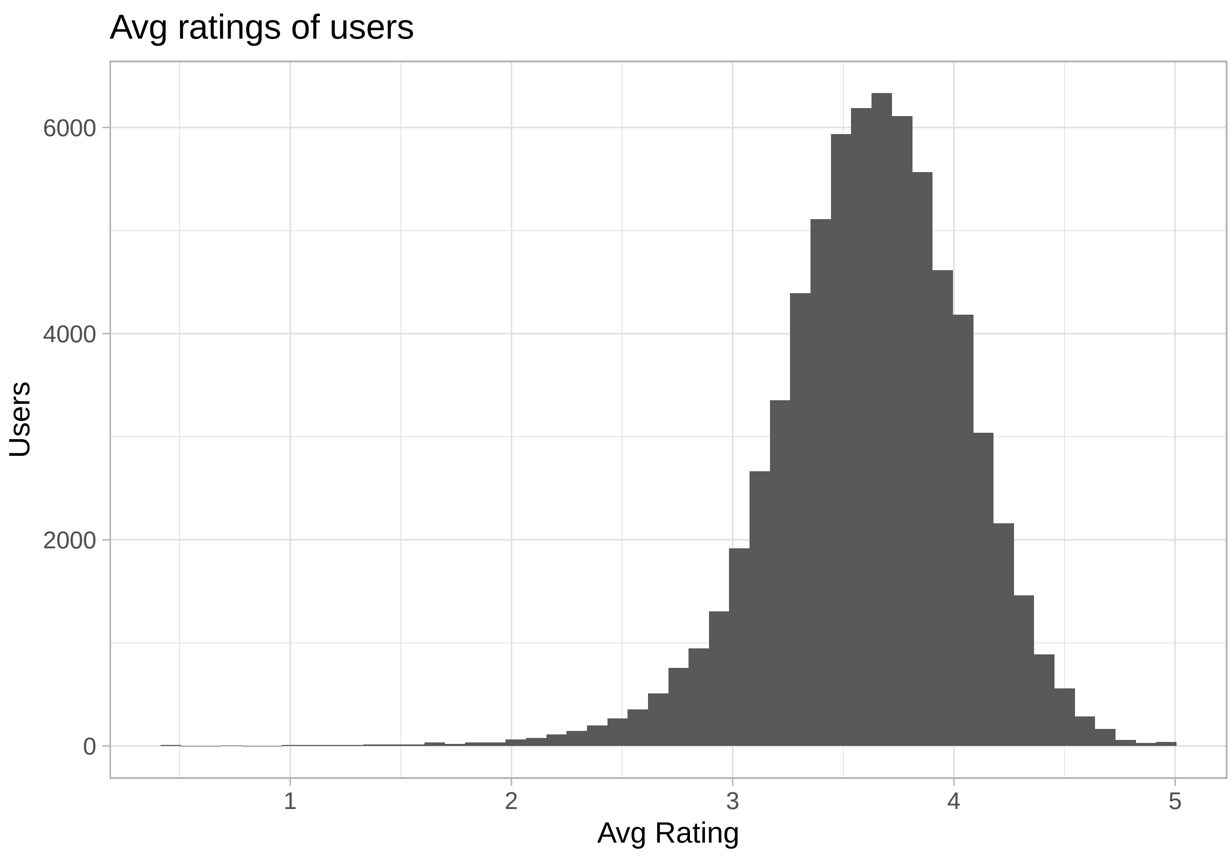
```
# When taken genre combinations into account, a similar picture is painted, but some
# entertainment genre combinations, notably a Action combination, make it higher up the list
# (e.g. Action/Drama/War, Action/Adventure)
```

```
# Average rating distribution of movies
edx %>%
  group_by(movieId) %>%
  summarise(avg_rating = sum(rating)/n()) %>%
  ggplot(aes(avg_rating)) +
    geom_histogram(bins=50) +
    theme_light() +
    labs(x="Avg Rating", y="Movies", title = "Number of ratings per movie")
```



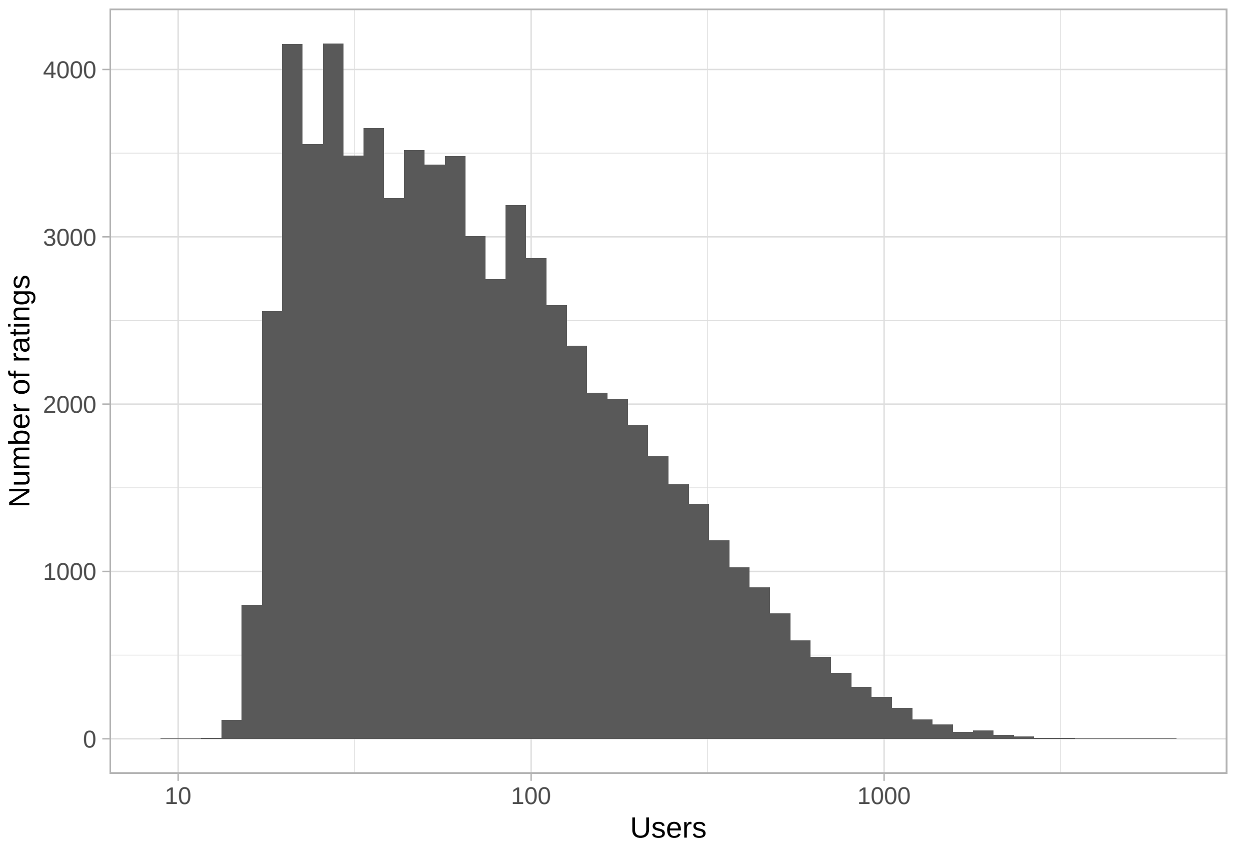
```
# Only very few movies get an average rating above about 4.2. Most average ratings
# are between 2.5 and 4.
```

```
# Average rating distribution of users
edx %>%
  group_by(userId) %>%
  summarise(avg_rating = sum(rating)/n()) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(bins=50) +
  theme_light() +
  labs(x="Avg Rating", y="Users", title = "Avg ratings of users")
```



```
# Most users rate movies between 3.5 and 4.2. This is higher than we've seen before on
# the average rating distribution of movies.
```

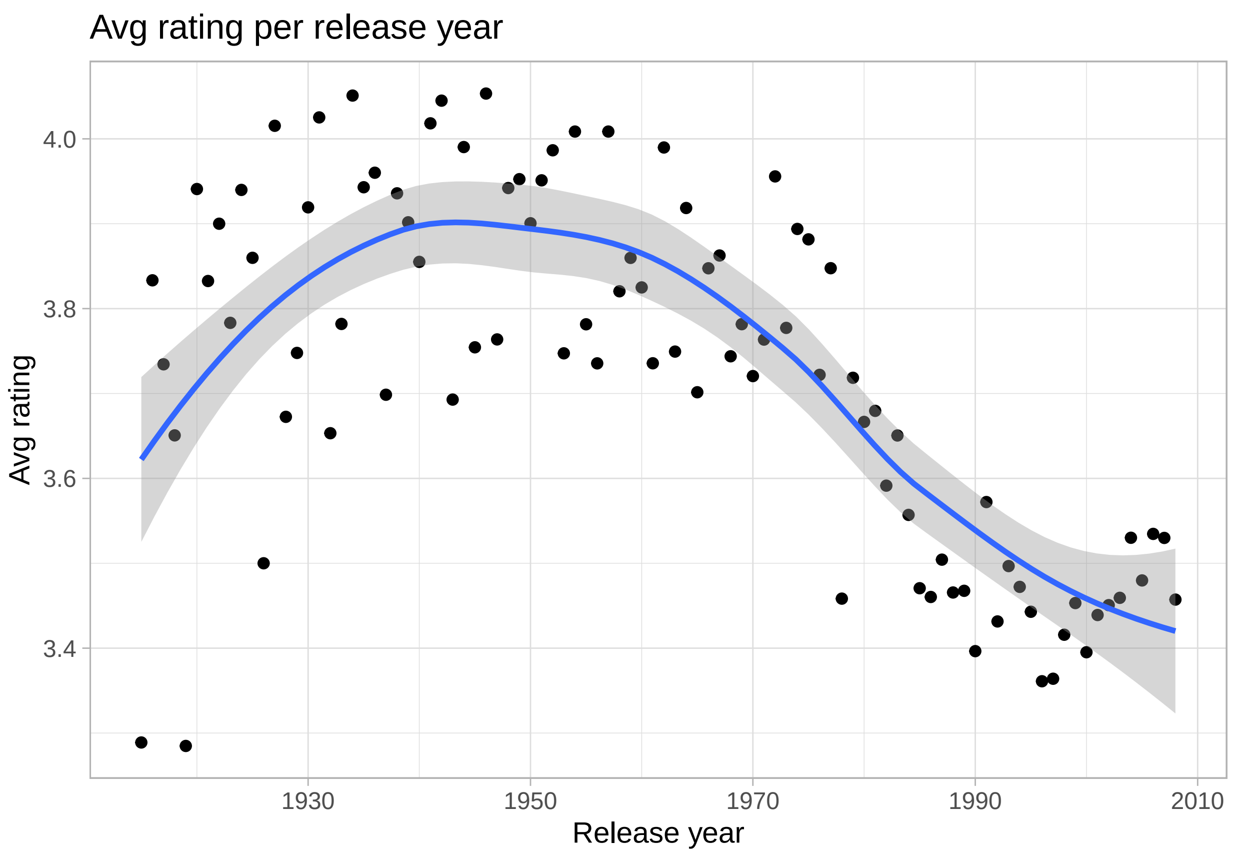
```
# Number of ratings per user
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
    geom_histogram( bins=50) +
    scale_x_log10() +
    #scale_y_log10() +  # log10 on number of ratings axis provides not a better readable graph
    theme_light() +
    labs(x = "Users", y = "Number of ratings")
```



```r
# Most users rate between a few and about 100 movies.
```

```r
# Average rating per release year with trendline
edx %>%
  group_by(releaseyear) %>%
  summarise(rating = mean(rating)) %>%
  ggplot(aes(releaseyear, rating)) +
    geom_point() +
    geom_smooth() +
    theme_light() +
    labs(x="Release year", y="Avg rating", title="Avg rating per release year")
```

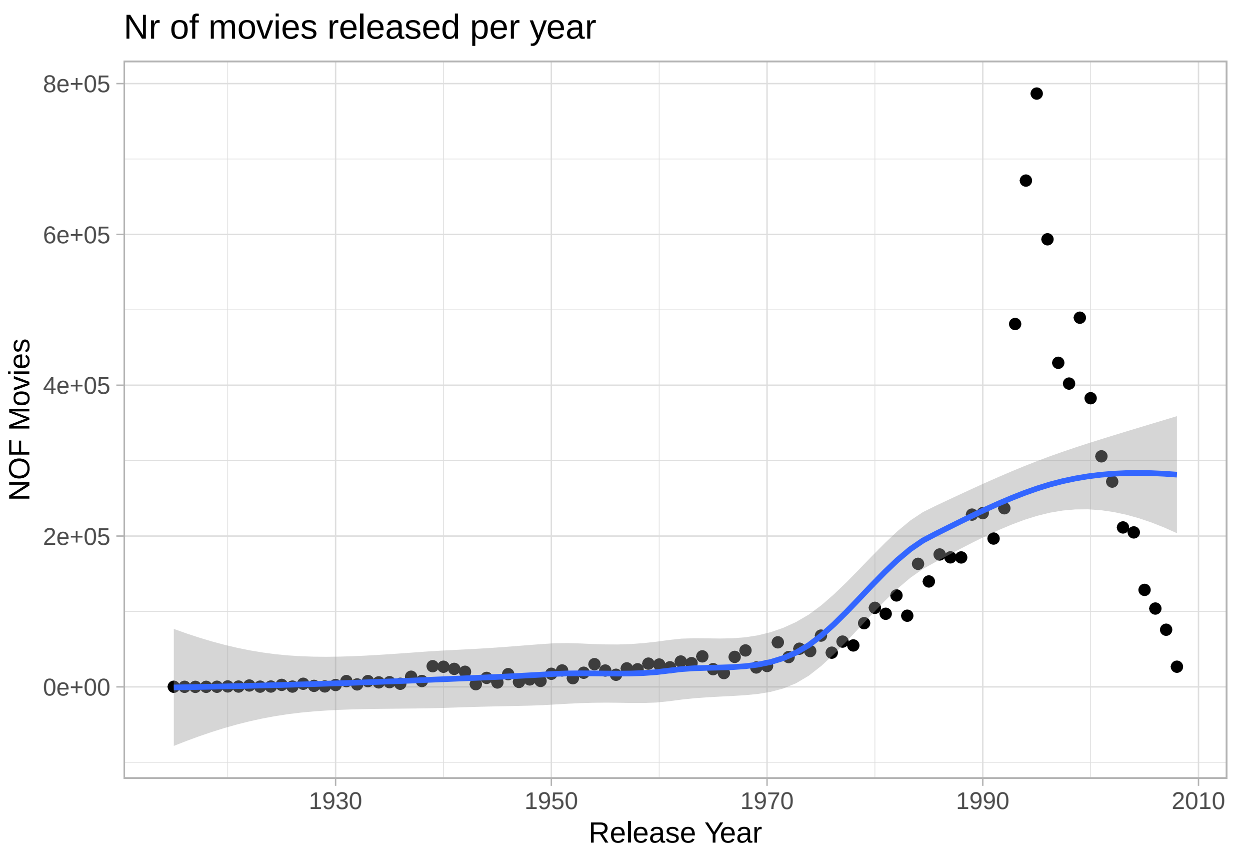
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



```
# Movies released before 1970 are generally rated higher than movies released in the
# last two decades. This is a know effect (only good stuff survive the test of time).
```

```
# Number of released movies per year
edx %>%
  group_by(releaseyear) %>%
  summarise(n=n()) %>%
  ggplot(aes(releaseyear, n)) +
    geom_point() +
    geom_smooth() +
    theme_light() +
    labs(x="Release Year", y="NOF Movies", title="Nr of movies released per year")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



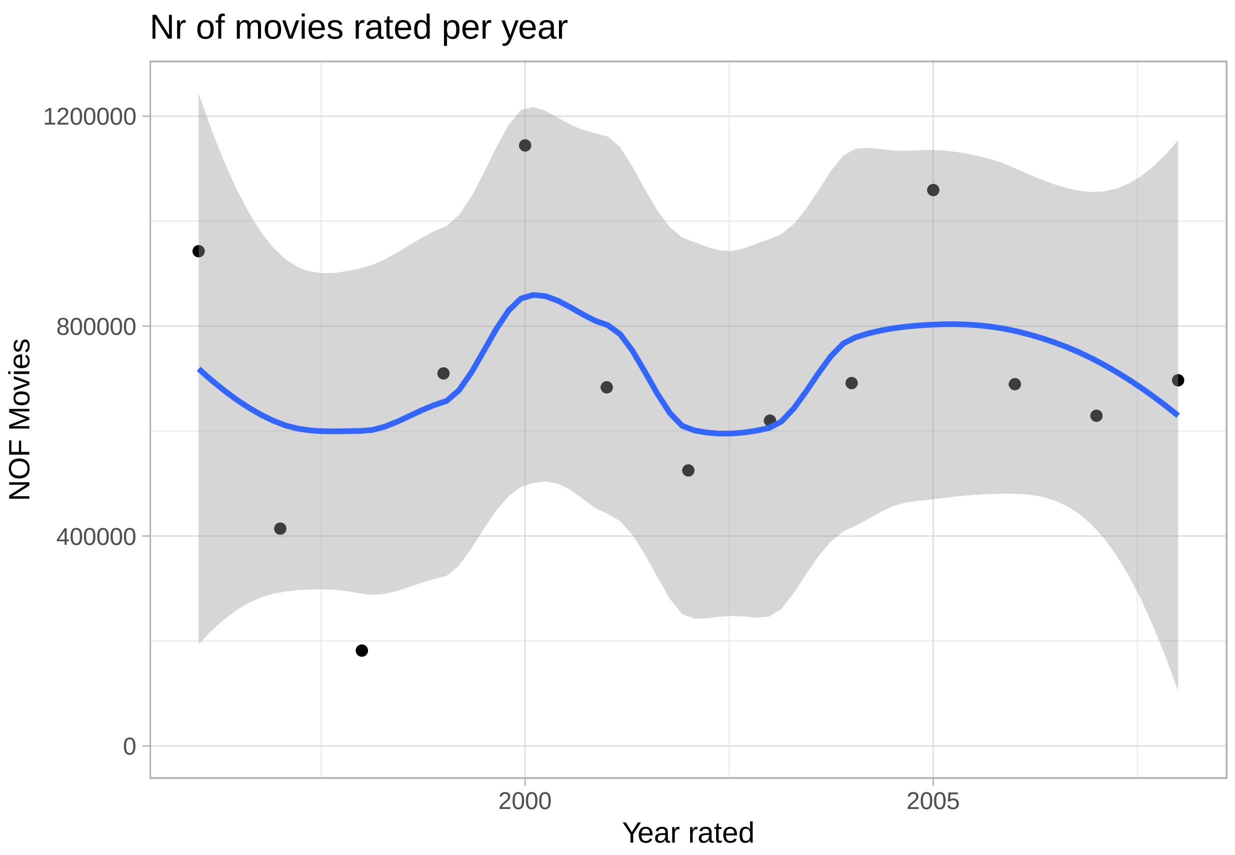
```
# Movie releases were relative stable before the 1970, then picked up and almost
# exploded 1990 and the following years, probably due to advancements in technology,
# production and market (e.g. movie theaters, movie rentals, tv...). Since before
# 2000 the number of movies released collapsed as quickly as its explosion a decade
# earlier.
```

```
# Number of ratings per year. Only include the years from 1996 to 2008, data before
# and after is 0.
edx %>%
  group_by(yearrated) %>%
  summarise(n=n()) %>%
  ggplot(aes(yearrated, n)) +
    geom_point() +
    geom_smooth() +
    theme_light() +
    xlim(1996, 2008) +
    labs(x="Year rated", y="NOF Movies", title="Nr of movies rated per year")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



```
# Movie ratings per year are stable with some outliers.
```

```
# average rating of movies of genres in 5-year bins
edx_5yr <- edx %>%
  mutate(five_year = floor((releaseyear - 1) / 5) * 5 + 1) %>%
  group_by(main_genre, five_year) %>%
  summarize(mean_rating = mean(rating, na.rm = TRUE)) %>%
  as.data.frame()
```

```
## `summarise()` has grouped output by 'main_genre'. You can override using the
## `.groups` argument.
```

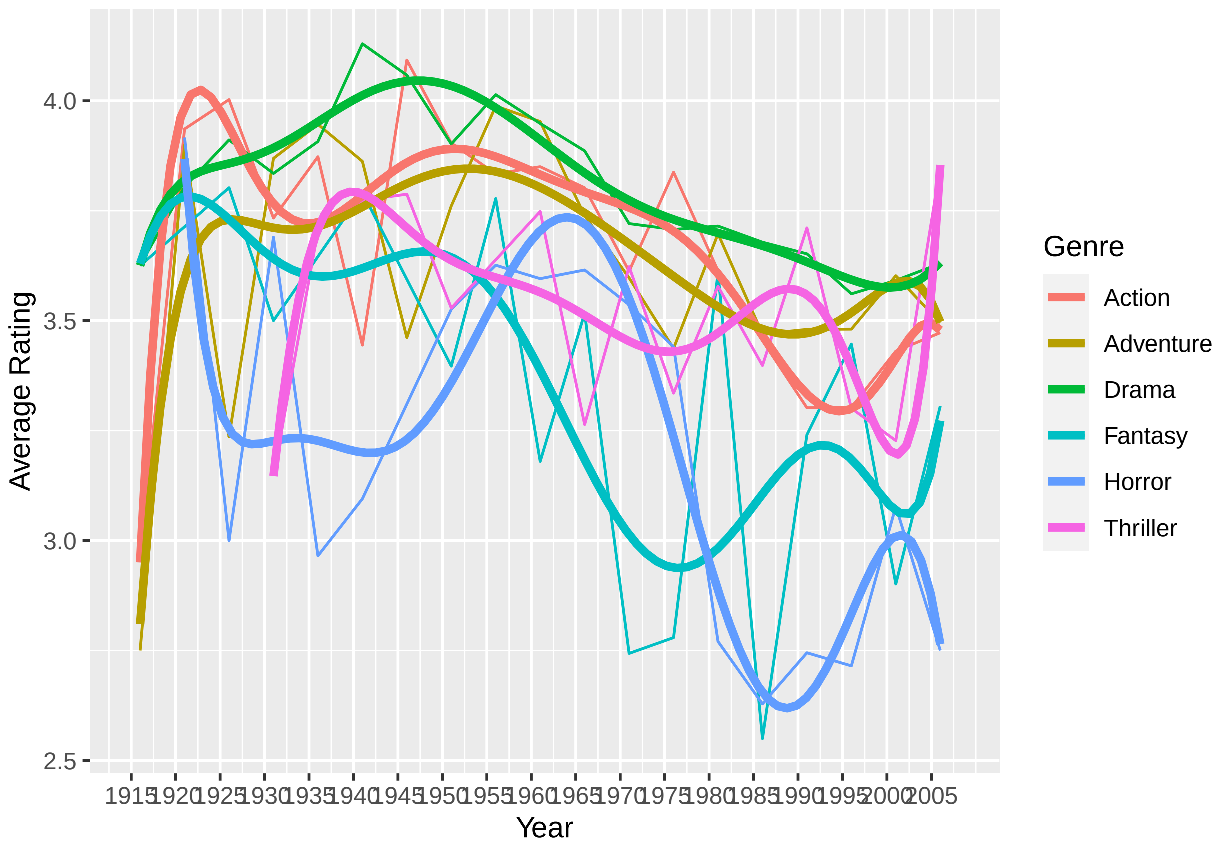
```
# include only some selected genres
edx_5yr_genre <- filter(edx_5yr, main_genre %in% c("Fantasy", "Action", "Adventure", "Drama", "Horror",

# plot the average rating of a genre with an overlayed fitting curve
ggplot(edx_5yr_genre, aes(x = five_year, y = mean_rating, color = main_genre)) +
  geom_line() +
  geom_smooth(method = "lm", formula = y ~ poly(x,8), se = FALSE, size = 1.5) +
  scale_x_continuous(limits = c(1915, 2008), breaks = seq(1915, 2008, 5)) +
  labs(x = "Year", y = "Average Rating", color = "Genre")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```

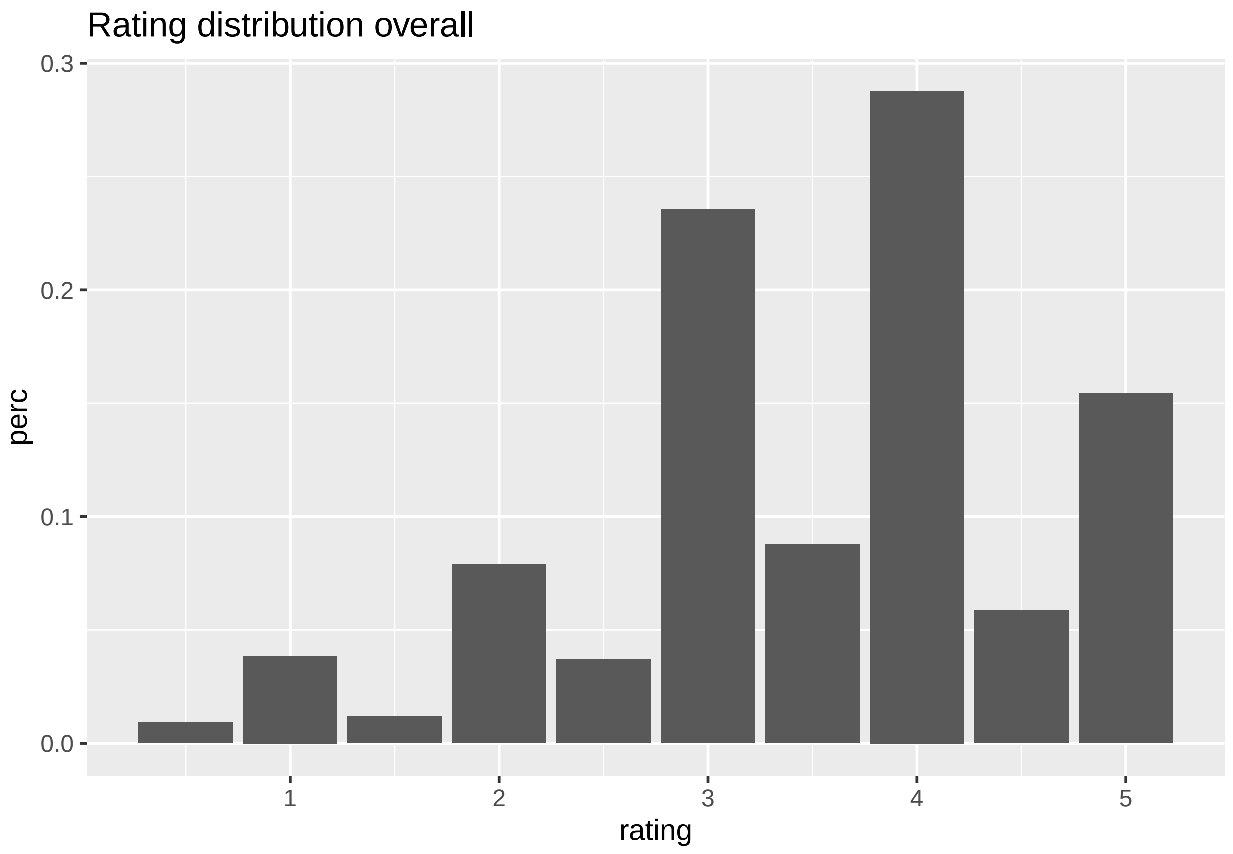


```
# some genres have pretty consistent average ratings over the years, others like e.g.
# Horror or Fantasy fluctuate a lot more.
```

Most rated genre:

Rating distribution per genre:

Data cleanup



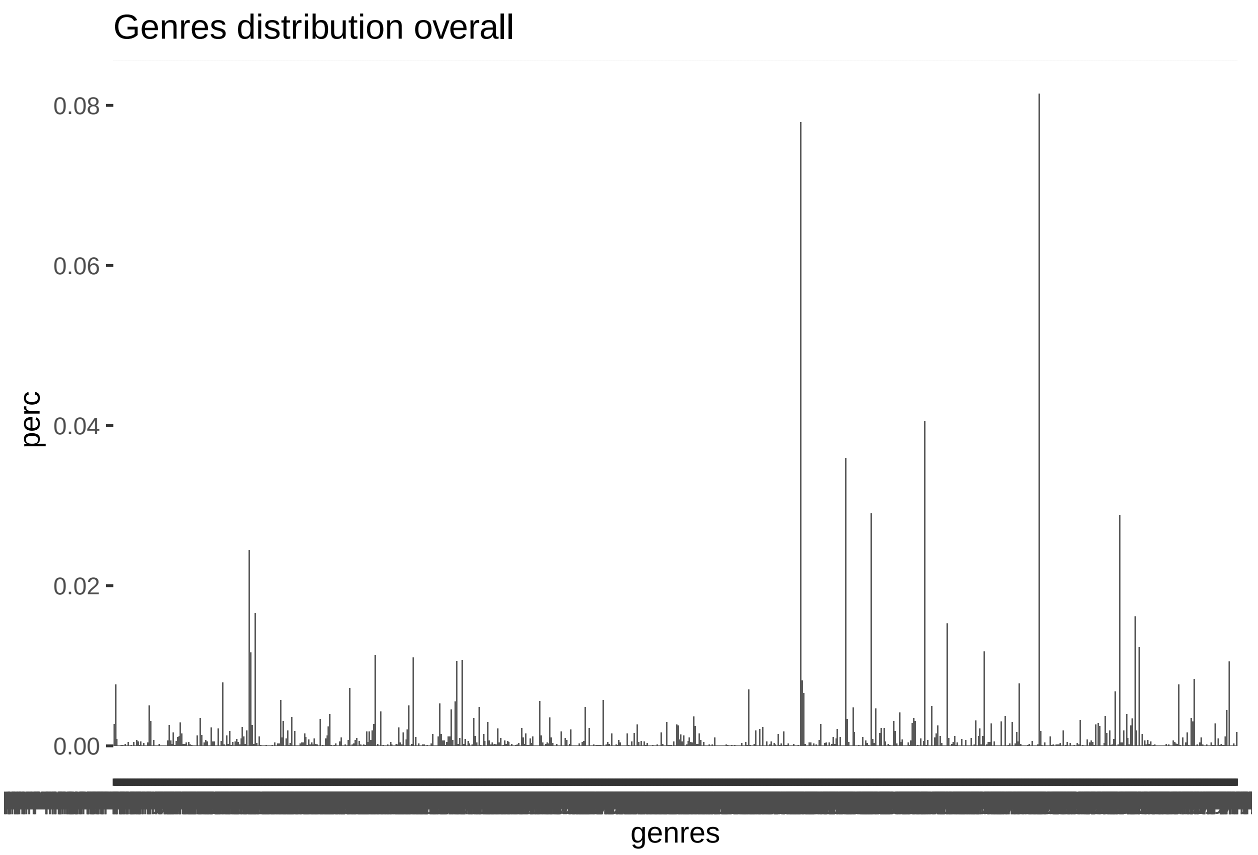
Is the rating of movies dependent of release year of the movie?

```r
#mutate()
```

Is the rating dependant of genre? (Of a user. e.g. UserX has 90% of rated movies in the genre of Comedy, he is more likely to rate a Comedy better than Action or Crime)

```r
#edx_genres <- edx %>% mutate(comedy = )

edx %>%
  group_by(genres) %>%
  summarise(perc = n()/nrow(edx)) %>%
    ggplot(aes(genres, perc)) +
    geom_col() +
    labs(title = "Genres distribution overall")
```



- check user ratings vs different genres
- . . .

Model

Results

RMSE

Conclusion

- summary
- limitations

Future Improvements

Further information about the users could imporve accuracy, e.g. shopping preferences, music taste, background infourmation like education level. But there privacy concern about the usage of user personal data has to be considered. Training with larger dataset would be beneficial but would require more capable systems (e.g. with GPU).

System

Hardware

All above computations are done with an . . . . CPU with . . . . and . . . .. of RAM.

```
#get_cpu()
#get_ram()
```

Software

This report is compiled using R markdown with RStudio.

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Arch Linux
##
## Matrix products: default
## BLAS:   /usr/lib/libblas.so.3.11.0
## LAPACK: /usr/lib/liblapack.so.3.11.0
##
## Random number generation:
##  RNG:     Mersenne-Twister
##  Normal:  Inversion
##  Sample:  Rounding
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] scales_1.2.1      lubridate_1.9.0   timechange_0.1.1  rmarkdown_2.18
##  [5] benchmarkme_1.0.8 devtools_2.4.5    usethis_2.1.6     caret_6.0-93
##  [9] lattice_0.20-45   forcats_0.5.2     stringr_1.4.1     dplyr_1.0.10
## [13] purrr_0.3.5       readr_2.1.3       tidyr_1.2.1       tibble_3.1.8
## [17] ggplot2_3.4.0     tidyverse_1.3.2
##
## loaded via a namespace (and not attached):
##   [1] googledrive_2.0.0     colorspace_2.0-3      ellipsis_0.3.2
##   [4] class_7.3-20          fs_1.5.2              rstudioapi_0.14
##   [7] farver_2.1.1          listenv_0.8.0         remotes_2.4.2
##  [10] bit64_4.0.5           prodlim_2019.11.13    fansi_1.0.3
##  [13] xml2_1.3.3            codetools_0.2-18      splines_4.2.2
##  [16] doParallel_1.0.17     cachem_1.0.6          knitr_1.41
##  [19] pkgload_1.3.2         jsonlite_1.8.3        pROC_1.18.0
##  [22] broom_1.0.1           dbplyr_2.2.1          shiny_1.7.4
##  [25] compiler_4.2.2        httr_1.4.4            backports_1.4.1
##  [28] assertthat_0.2.1      Matrix_1.5-1          fastmap_1.1.0
##  [31] gargle_1.2.1          cli_3.4.1             later_1.3.0
##  [34] prettyunits_1.1.1     htmltools_0.5.4       tools_4.2.2
##  [37] gtable_0.3.1         glue_1.6.2           reshape2_1.4.4
##  [40] Rcpp_1.0.9           cellranger_1.1.0     vctrs_0.5.1
##  [43] nlme_3.1-160         iterators_1.0.14     timeDate_4021.106
##  [46] gower_1.0.0          xfun_0.35            globals_0.16.2
##  [49] ps_1.7.2             rvest_1.0.3          mime_0.12
##  [52] miniUI_0.1.1.1       lifecycle_1.0.3      googlesheets4_1.0.1
##  [55] future_1.29.0        MASS_7.3-58.1        ipred_0.9-13
##  [58] vroom_1.6.0          hms_1.1.2            promises_1.2.0.1
##  [61] yaml_2.3.6           memoise_2.0.1        rpart_4.1.19
##  [64] stringi_1.7.8        highr_0.9            foreach_1.5.2
##  [67] hardhat_1.2.0        pkgbuild_1.4.0       benchmarkmeData_1.0.4
##  [70] lava_1.7.0           rlang_1.0.6          pkgconfig_2.0.3
##  [73] evaluate_0.18        labeling_0.4.2       htmlwidgets_1.6.1
##  [76] recipes_1.0.3        bit_4.0.5            tidyselect_1.2.0
##  [79] processx_3.8.0       parallelly_1.32.1    plyr_1.8.8
##  [82] magrittr_2.0.3       R6_2.5.1             profvis_0.3.7
##  [85] generics_0.1.3       DBI_1.1.3            mgcv_1.8-41
##  [88] pillar_1.8.1         haven_2.5.1          withr_2.5.0
##  [91] survival_3.4-0       nnet_7.3-18          future.apply_1.10.0
##  [94] modelr_0.1.10        crayon_1.5.2         utf8_1.2.2
##  [97] urlchecker_1.0.1     tzdb_0.3.0           grid_4.2.2
## [100] readxl_1.4.1         data.table_1.14.6    callr_3.7.3
## [103] ModelMetrics_1.2.2.2 reprex_2.0.2         digest_0.6.30
## [106] xtable_1.8-4         httpuv_1.6.8         stats4_4.2.2
## [109] munsell_0.5.0        sessioninfo_1.2.2
```

Resources

[1] Rafael Irizarry. 2018. Introduction to Data Science. https://rafalab.dfci.harvard.edu/dsbook/

Harper, F. Maxwell, and Joseph A. Konstan. 2015. "The MovieLens Datasets." *ACM Transactions on Interactive Intelligent Systems* 5 (4): 1–19. https://doi.org/10.1145/2827872.